UM2031
用户手册

STM32Cube的声级计库软件扩展

引言

声级计库软件用户手册描述了SoundMeterR（SMR）模块配置及其接口。

该用户手册描述了如何将模块集成到主程序，例如X-CUBE-AUDIO扩展软件中。它还提供了对基础算法的基本理解。

声级计库软件扩展适用于16或32位输入/输出格式。它是X-CUBE-AUDIO固件包的一部分。

目录

表格索引

图片索引

1 模块概述

1.1 算法功能

SoundMeteR（SMR）模块负责在对数尺度上测量传入信号的电平。它以尺度转换、平滑滤波和加权滤波为基础。

当前实现对所有计算使用32位分辨率，可与16或32位输入/输出格式一起使用。支持的采样率为8 kHz、16 kHz和48 kHz。

1.2 模块配置

SMR模块支持单声道和立体声16位或32位I/O数据。其存储器受限于960个采样的最大输入帧大小，相当于10 ms的48 kHz立体声信号。

根据I/O格式、Cortex®内核和使用的工具链，有多个模块版本可供使用：

- SMR_CM4_IAR.a / SMR_CM4_GCC.a / SMR_CM4_Keil.lib：适用于16位输入/输出缓冲区，在任何使用包含Cortex®-M4指令集的内核的STM32微控制器上运行。
- SMR_32b_CM4_IAR.a / SMR_32b_CM4_GCC.a / SMR_32b _CM4_Keil.lib：适用于32位输入/输出缓冲区，在任何使用包含Cortex®-M4指令集的内核的STM32微控制器上运行。
- SMR_CM7_IAR.a / SMR_CM7_GCC.a / SMR_CM7_Keil.lib：适用于16位输入/输出缓冲区，在任何使用包含Cortex®-M7指令集的内核的STM32微控制器上运行。
- SMR_32b_CM7_IAR.a / SMR_32b_CM7_GCC.a / SMR_32b _CM7_Keil.lib：适用于32位输入/输出缓冲区，在任何使用包含Cortex®-M7指令集的内核的STM32微控制器上运行。

1.3 资源总结

*表 1*包含模块对存储器和频率（MHz）的要求。

使用IAR Embedded Workbench for ARM V7.40 （IAR Embedded Workbench common components v7.2）测量板上的内存占用量。

表1. 资源总结

I/O	Core	Flash code (.text)	Flash data (.rodata)	Stack	Persistent RAM	Scratch RAM(1)	Frequency (MHz)
16 位	M4	4502	8	80	392	3844	5.8
16 位	M7	4520	8				3.9
[32 位]	M4	4208	8				5.5
[32 位]	M7	4624	8				3.8

1. Scratch RAM是指与其它同一优先级进程共享的内存。

注： *此表所示内存使用情况是在上STM32F7板上测量所得，且堆栈放置于DTCM存储器中。*

Scratch RAM是指与其它同一优先级进程共享的内存,SMR例程使用此内存时并不是从一帧到另外一帧。

2 模块接口

使用SRM模块时需要集成其两个文件：SMR_xxx_My.a/.lib库和smr_glo.h头文件。它们包含所有要导出到软件集成框架的定义和结构。

注： *audio_fw_glo.h文件是通用头文件，它对于所有音频模块都通用且必须包含在音频框架中。*

2.1 API

六个通用函数软件接口：

- smr_reset函数
- smr_setParam函数
- smr_getParam函数
- smr_setConfig函数
- smr_getConfig函数
- smr_process函数

下面几节将分别描述每一个函数。

2.1.1 smr_reset函数

该例程初始化SMR模块的persistent memory，并使用默认值初始化静态和动态参数。

int32_t smr_reset(void *persistent_mem_ptr, void *scratch_mem_ptr);

表2. smr_reset

I/O	名称	类型	说明
输入	persistent_mem_ptr	void *	指向内部persistent memory的指针
输入	scratch_mem_ptr	void *	指向内部scratch memory的指针
返回值	-	int32_t	错误值

当实时处理尚未开始时，在初始化阶段，必须至少调用一次该例程。

2.1.2 smr_setParam函数

该例程从主框架将模块静态参数写入模块内存。可以在复位例程后，实时处理开始前对其进行调用。它负责处理静态参数（即模块处理期间参数值不能更改的参数）。

int32_t smr_setParam(smr_static_param_t *input_static_param_ptr, void *persistent_mem_ptr);

表3. smr_setParam

I/O	名称	类型	说明
输入	input_static_param_ptr	smr_static_param_t*	指向静态参数结构的指针
输入	persistent_mem_ptr	void *	指向内部persistent memory的指针
返回值	-	int32_t	错误值

2.1.3 smr_getParam函数

该例程从模块内存为主框架获取模块静态参数。可以在复位例程后，实时处理开始前对其进行调用。它负责处理静态参数（即模块处理期间参数值不能更改的参数）。

int32_t smr_getParam(smr_static_param_t *input_static_param_ptr, void *persistent_mem_ptr);

表4. smr_getParam

I/O	名称	类型	说明
输入	input_static_param_ptr	smr_static_param_t *	指向静态参数结构的指针
输入	persistent_mem_ptr	void *	指向内部persistent memory的指针
返回值	-	int32_t	错误值

2.1.4 smr_setConfig函数

该例程从主框架为模块内存设置模块动态参数。可以在处理期间的任何时间对其进行调用。

int32_t smr_setConfig(smr_dynamic_param_t *input_dynamic_param_ptr, void *persistent_mem_ptr);

表5. smr_setConfig

I/O	名称	类型	说明
输入	input_dynamic_param_ptr	smr_dynamic_param_t *	指向动态参数结构的指针
输入	persistent_mem_ptr	void *	指向内部persistent memory的指针
返回值	-	int32_t	错误值

2.1.5 smr_getConfig函数

该例程从内部persistent memory为主框架获取模块动态参数。可以在处理期间的任何时间对其进行调用。

int32_t smr_getConfig(smr_dynamic_param_t *input_dynamic_param_ptr, void *static_mem_ptr);

表6. smr_getConfig

I/O	名称	类型	说明
输入	input_dynamic_param_ptr	smr_dynamic_param_t *	指向动态参数结构的指针
输入	persistent_mem_ptr	void *	指向内部persistent memory的指针
返回值	-	int32_t	错误值

2.1.6 smr_process函数

该例程是模块的主要处理例程。

随时可以被调用来处理每帧数据。

int32_t smr_process(buffer_t *input_buffer, buffer_t *output_buffer, void *persistent_mem_ptr);

表7. smr_process

I/O	名称	类型	说明
输入	input_buffer	buffer_t *	指向输入缓冲区结构的指针
输出	output_buffer	buffer_t *	指向输出缓冲区结构的指针
输入	persistent_mem_ptr	void *	指向内部persistent memory的指针
返回值	-	int32_t	错误值

此处理例程可以在适当的时候运行，这意味着同一缓冲区可以同时用于输入和输出。

2.2 外部定义和类型

2.2.1 输入和输出缓冲区

SMR库使用扩展I/O缓冲区，除了采样，它还包含一些关于数据流的有用信息，例如通道数、每个采样的字节数和交叉模式。

每次调用处理例程之前，必须遵循并填写I/O缓冲区结构类型（如下文所述），否则将返回错误：

```
typedef struct {
   int32_t    nb_channels;
   int32_t    nb_bytes_per_Sample;
   void       *data_ptr;
   int32_t    buffer_size;
   int32_t    mode;
} buffer_t;
```

表8. 输入和输出缓冲区

名称	类型	说明
nb_channels	int32_t	数据中的通道数：1表示单声道，2表示立体声
nb_bytes_per_Sample	int32_t	16位 = 2，32位 = 4，SMR支持16位和32位格式的音频采样。
data_ptr	void *	指向数据缓冲区的指针（必须由主框架分配）
buffer_size	int32_t	数据缓冲区中每个通道的采样数
模式	int32_t	如果是立体声数据流，则左右声道可以是交叉模式。 0 = 不交叉，1 = 交叉。 SMR模块仅支持交叉模式。

2.2.2 返回的错误值

*表 9*列出了可能返回的错误值：

表9. 返回的错误值

定义	值	说明
SMR_ERROR_NONE	0	未检测到错误
SMR_UNSUPPORTED_INTERLEAVING_MODE	-1	如果输入数据不交叉
SMR_UNSUPPORTED_NUMBER_OF_CHANNELS	-2	输入数据既不是单声道也不是立体声
SMR_UNSUPPORTED_NUMBER_OF_BYTEPERSAMPLE	-3	输入数据既不是16位也不是32位采样格式
SMR_UNSUPPORTED_AVERAGING_TIME	-4	averaging_time不在以下范围内：[0: 10000]
SMR_UNSUPPORTED_FILTER_TYPE	-5	filter_type不在支持的类型列表中
SMR_UNSUPPORTED_SAMPLING_RATE	-6	sampling_rate不等于8000、16000或48000
SMR_BAD_HW	-7	库不支持的硬件

2.3 静态参数结构

在调用smr_setParam()函数之前，使用相应的静态参数结构设置SMR初始参数。

```
struct smr_static_param {
    int32_t sampling rate;
}
typedef struct smr_static_param smr_static_param_t;
```

表10. 静态参数结构

名称	类型	说明
sampling_rate	int32_t	输入缓冲区采样率（Hz）。只支持8 kHz、16 kHz和48 kHz。该值用于计算一些时间常量以及初始化滤波系数。

2.4 动态参数结构

SMR库在其动态参数结构中推荐输入和输出参数。

对于输入参数，可通过在调用smr_setConfig()函数之前设置动态参数结构来设置或更改SMR配置。

对于输出参数，必须在访问更新的输出动态参数之前调用smr_getConfig()函数。

输入和输出参数的说明如下：

```
struct smr_dynamic_param {
    int32_t enable;              /* input variable  */
    int16_t averaging_time;      /* input variable  */
```

```
    int16_t filter_type;          /* input variable  */
    int32_t mean_level_left;      /* output variable  */
    int32_t mean_level_right;     /* output variable  */
}
```

表11. 动态参数结构

名称	类型	说明
启用	int32_t	1 = 使能SMR的处理。 0 = 禁用SMR的处理，SMR输出电平值不可靠。
averaging_time	int16_t	用来平滑通过算法计算得出的瞬时电平的时间常量。 它以ms为单位。最小值为0 ms，最大值为10,000 ms。
filter_type	int16_t	电平测量之前的预滤波滤波器。有4种设置可用： #define SMR_PREFILTER_NONE 0 = 无预滤波器 #define SMR_PREFILTER_AWEIGHTING 1 = A加权预滤波器 #define SMR_PREFILTER_CWEIGHTING 2 = C加权预滤波器 #define SMR_PREFILTER_DCREMOVE 3 = 直流消除预滤波器
mean_level_left	int32_t	左声道（立体声输入）或单声道的声级。输出格式为Q29.2格式。
mean_level_right	int32_t	右声道的声级。当输入为单声道时，该值无关紧要。输出格式为Q29.2格式。

3 算法描述

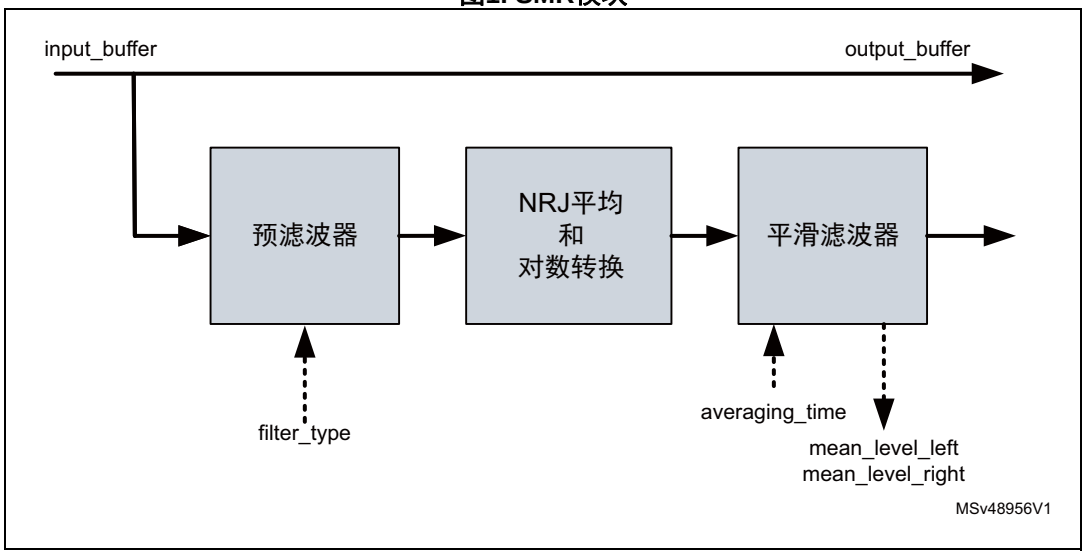
3.1 处理步骤

SMR模块推荐对输入信号在对数尺度上进行测量。它取16或32位、单声道或立体声音频流信号作为输入，并从对信号使用预滤波器（如已配置）开始。然后，它计算当前帧的能量，并将其转换至对数尺度。其次，通过由平均时间参数配置的平滑滤波器传输该电平。最后，更新持久存储结构中的输出电平，通过调用smr_getConfig()函数和动态参数结构可以访问该输出电平。

SMR是全通模块，这意味着输出缓冲区是输入缓冲区的精确复制。

*图 1*所示为SMR模块。

图1. SMR模块



3.2 数据格式

SMR模块的输入应为音频流，单声道或立体声，16或32位格式。运行算法的最大缓冲区大小为总共960个采样。例如，对于48 kHz采样率的立体声信号，这相当于10 ms。以32位分辨率执行所有操作。输出格式与输入缓冲区的相同。

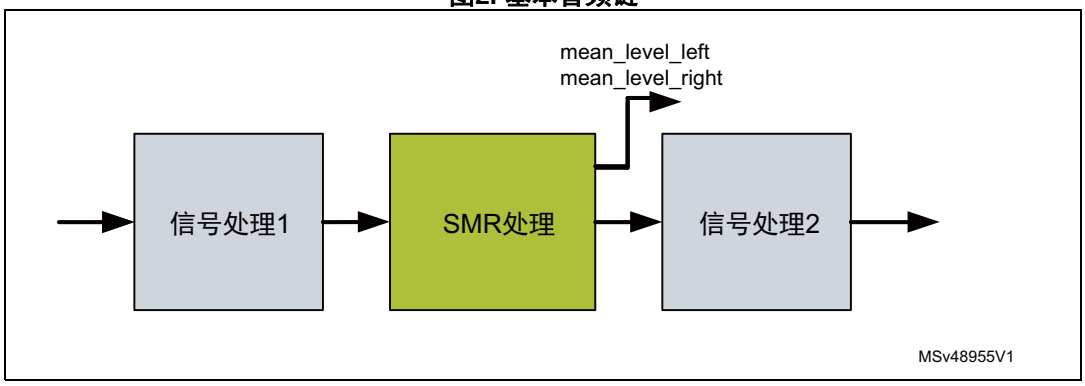
4 应用描述

SMR库可在Cortex® M4或Cortex® M7内核上运行。它们分别可以在STM32F4、STM32L4、STM32F7或STM32H7系列微控制器上集成和运行。不依赖其他硬件。

4.1 最佳设置建议

SMR模块是全通模块，可在音频处理链的任何位置执行。应将其置于用户想要获取信号电平测量值的位置。请参见*图 2：基本音频链*。

图2. 基本音频链



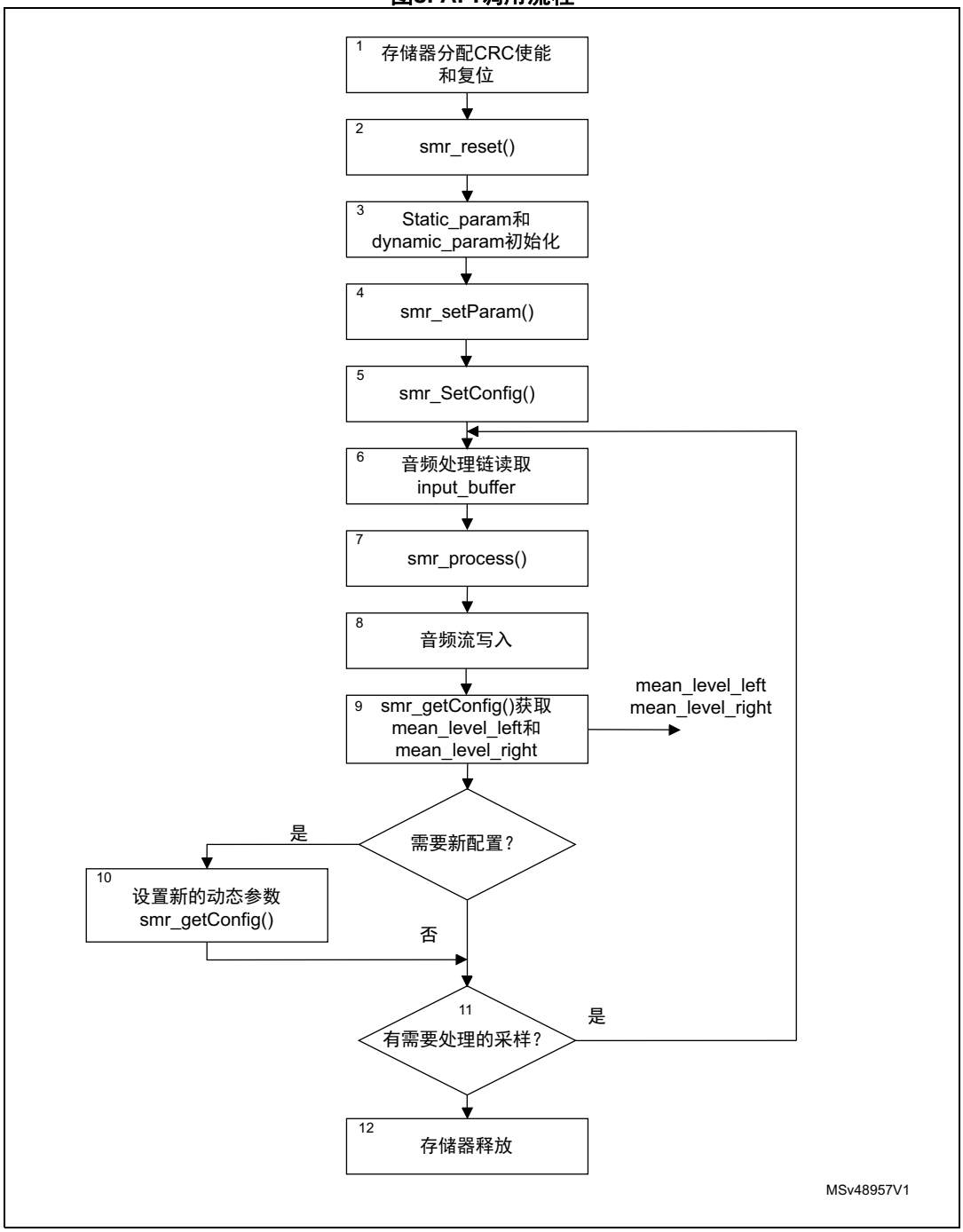
4.1.1 模块集成示例

STM32F746G探索板和STM32F469I探索板上提供了Cube扩展SMR集成示例。更多详细信息，请参考提供的集成代码。

4.1.2 模块API调用

*图 3*显示了API调用序列。

图3. API调用流程



1. 如上文所述，必须分配SMR scratch memory和persistent memory，以及输入和输出缓冲区。
2. 一旦分配了存储空间，对smr_reset()函数的调用将初始化内部变量。
3. 此时可以设置SMR静态和动态参数结构。
4. 调用smr_setParam()例程应用静态参数。
5. 调用smr_setConfig()函数应用动态参数。
6. 从音频接口读取音频流，并且必须按照音频流特性（通道数、采样率、交叉和数据指针）填充input_buffer结构。此外，还必须设置输出缓冲区结构。
7. 调用smr_process()函数将执行SMR算法。
8. 此时，可以在合适的接口中写入输出音频流。对于SMR，在任何情况下输出缓冲区都与输入缓冲区相同。
9. 为了获得更新的SMR mean_level_left和mean_level_right变量，必须在访问动态参数结构之前调用smr_getConfig()函数。
10. 如果需要，用户可以设置新的动态参数，并调用smr_setConfig()函数更新模块配置。
11. 如果应用仍在运行且有新的输入采样需要处理，则回到第6步，否则处理循环结束。
12. 一旦处理循环结束，必须释放分配的存储空间。

5 如何调整和运行应用

SMR的配置参数很少。但是，应根据用户要测量的对象进行调整。

5.1 averaging_time：

必须根据用户需要的精度调整平均时间。平均时间过短可导致电平回读的不稳定性过高，而平均时间过长则可能隐藏传入信号的突升或突降。

声级计通常提供“快速”测量和“慢速”测量，前者的平均时间常量为125 ms，后者为1 s。

图4. 基本音频链

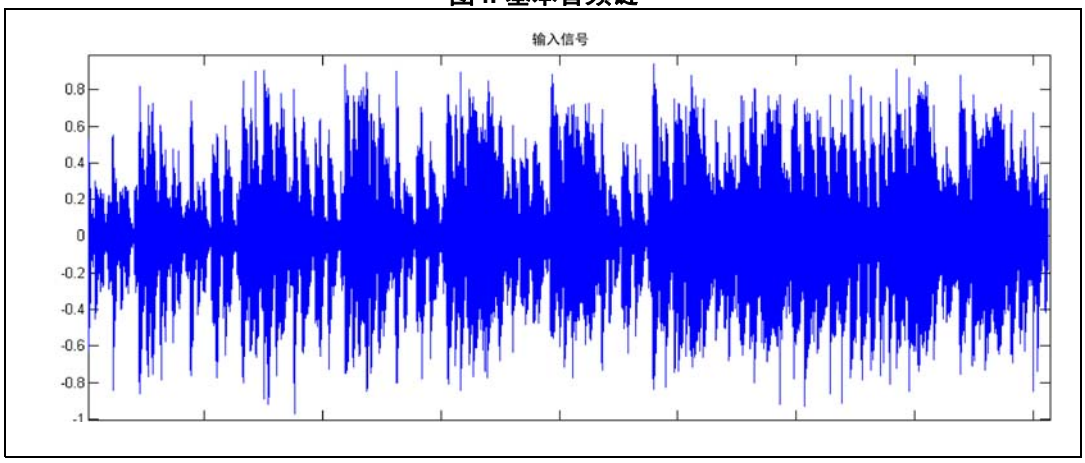
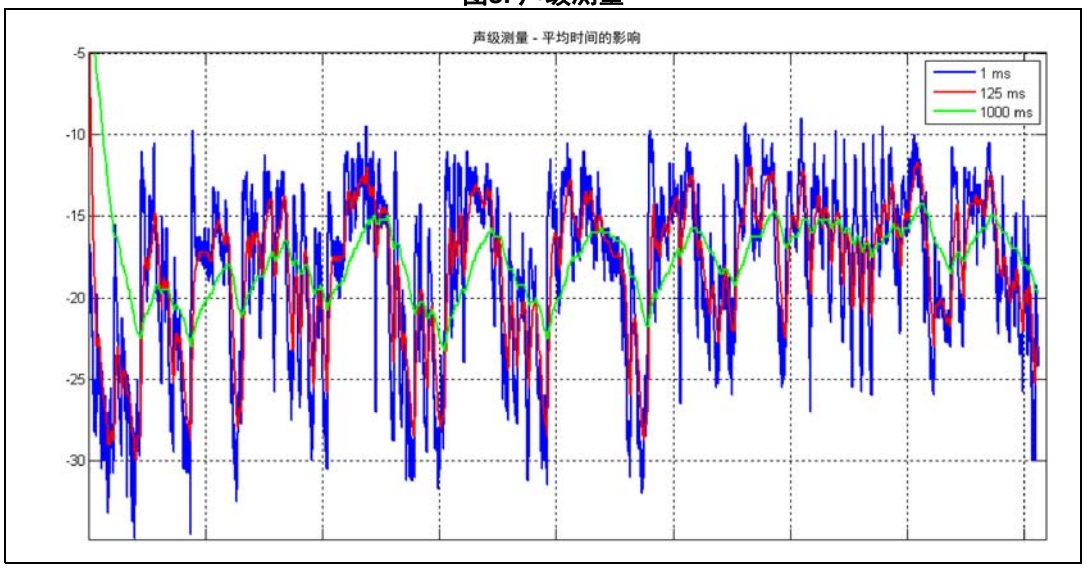


图5. 声级测量



5.2 filter_type：

当用户想要获取信号频谱无任何加权的测量值时，应使用直流消除预滤波器，除非为低通滤波器设置极低的截止频率。

A加权滤波器逼近低声级时的反向等响度曲线，而C加权适用于高声级。如今最常用的是A加权滤波器，因为它与主观测试的相关性良好。

6 版本历史

表12. 文档版本历史

日期	版本	变更
2017年1月23日	1	初始版本。

表13. 中文文档版本历史

日期	版本	变更
2018年12月11日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。